**MarketHub**

*A Project Report Submitted*

to

**MANIPAL ACADEMY OF HIGHER EDUCATION**

*for Partial Fulfilment of the Requirement for the*

*Award of the Degree*

*Of*

**Bachelor of Technology**

*in*

**Computer and Communication Engineering**

*by*

**Shravani Sanjay Sawant, Aditi Rishiraj**

**230953006, 230953011**

*Under the guidance of*

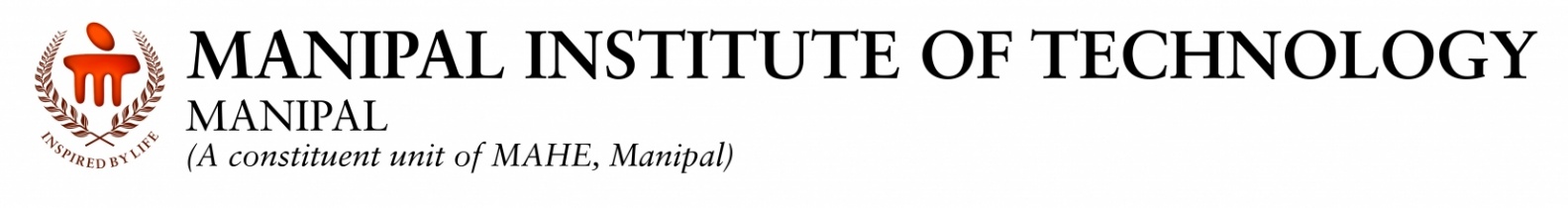
Dr. Diana Olivia

Associate Professor

Department of I&CT

Manipal Institute of Technology

Manipal, Karnataka, India



Mrs. Swathi B P

Assistant Professor -Senior Scale

Department of I&CT

Manipal Institute of Technology

Manipal, Karnataka, India

**April 2025**

**ABSTRACT**

MarketHub is an integrated web-based platform designed to streamline wholesale business operations by providing tools for suppliers and customers to manage purchasing, transport, shipment tracking, and returns efficiently. The system leverages Python Flask for backend development and MySQL for database management. Key features include user management, product cataloguing, order tracking, shipment management, discounts/offers management, returns/refunds handling, and warehouse monitoring. This report outlines the development process, technical specifications, and implementation details of **MarketHub**.

**ACM taxonomy**

* Information Systems: E-commerce Infrastructure
* Information systems → Relational database model;
* Information systems → Entity relationship models;
* Information systems → Query optimization;
* Information systems → Join algorithms;
* Information systems → Query planning;
* Information systems → Data exchange;
* Information systems → Deduplication;
* Information systems → Universal Description Discovery and Integration (UDDI).
* Software Engineering: Requirements/Specifications

**SDG**

MarketHub aligns with several Sustainable Development Goals:

* Goal 8: Decent Work and Economic Growth by promoting efficient wholesale business operations.
* Goal 9: Industry, Innovation, and Infrastructure by leveraging technology for streamlined business processes.

**Table of Contents-Aditi**

**(Final formatting)-Aditi**

**List of Tables-Aditi**

- Table 1: Relational Schema for User Table

- Table 2: Relational Schema for Supplier Table

- Table 3: Relational Schema for Customer Table

**List of Figures-Aditi**

* Figure 1: Entity-Relationship Diagram (ERD) for MarketHub
* Figure 2: System Architecture Block Diagram

**Abbreviations**

- RBAC: Role-Based Access Control

- SRS: Software Requirements Specification

- ERD: Entity-Relationship Diagram

**Chapter 1**

**Introduction**

**1.1 Purpose**

MarketHub is made to help wholesale businesses work more easily and efficiently. It’s a digital platform where suppliers and customers can connect and manage everything in one place. The main goal is to make everyday tasks like buying products, tracking shipments, handling returns, and managing stock faster and simpler.

MarketHub is useful for both businesses and customers. Suppliers can list their products, give discounts, and track stock levels, while customers can browse, place orders, and see where their shipments are. The platform also makes it easier to return items or ask for refunds if something goes wrong.

Another big purpose of MarketHub is to save time and reduce mistakes by automating tasks that would normally be done by hand. It’s designed to be easy to use, so people with different roles like warehouse managers, delivery staff, or business owners can quickly find what they need and get their work done. MarketHub also helps businesses make better decisions by showing useful data, like which products are selling the most or where delays are happening.

Overall, MarketHub helps businesses grow by improving how they serve their customers and manage their daily operations.

**1.2 Scope**

MarketHub includes several key modules:

* User Management: Registration, login, and role-based access control.
* Product Management: Viewing, categorizing, and reviewing products.
* Order Management: Order placement, tracking, and order history.
* Shipment Management: Transport assignment, status tracking, delivery estimation.
* Discounts and Offers: Managing discounts and offers for products.
* Returns and Refunds: Handling product returns and refunds, as well as complaint resolution.
* Warehouse Management: Stock monitoring and restock alerts.

**1.3 Intended Audience**

This document is intended for:

* Developers: For the technical implementation of the system.
* Testers: These are used to understand the functionalities and performance requirements.
* Stakeholders: To review the system's functionality, scope, and features.

**Chapter 2**

**Background**

In today’s world, many wholesale businesses use online platforms to manage their work. However, a lot of these platforms are not well-connected or complete. This means that suppliers and customers often face problems while trying to place orders, track shipments, or manage returns. For example, a customer might not know when their order will arrive, or a supplier might lose track of which products have been shipped and which haven’t. These small problems can add up and create delays, confusion, and unhappy customers.

One of the main issues with existing systems is that they only focus on certain parts of the business, instead of bringing everything together. Many platforms are designed for retail businesses where people buy single items for personal use. These platforms usually don’t work well for wholesale businesses that deal with larger orders, bulk shipments, and more complex processes.

This is where **MarketHub** comes in. It was created to fix these common problems. MarketHub connects all parts of a wholesale business—from suppliers to customers—and helps them work together smoothly. It combines different tools like order placement, product management, shipment tracking, warehouse control, returns handling, and customer communication into one single platform. This makes it easier for everyone involved to stay informed and get their work done quickly and correctly.

By using MarketHub, businesses don’t have to switch between different apps or websites. Everything they need is in one place. This saves time, reduces mistakes, and makes customers happier. It also helps businesses grow by making their operations more organized and efficient.

In short, MarketHub was made to improve the way wholesale businesses work by solving problems that other platforms don’t fully handle. It gives both suppliers and customers a better, faster, and more reliable way to do business.

**Chapter 3**

**Objective**

**3.1 Problem Statement**

Many wholesale businesses today face several problems in how they run their daily activities. One big issue is that order management is often slow and confusing. Sometimes, it takes a long time to process orders, and customers are left waiting without knowing what’s going on. Another problem is the lack of real-time tracking. Once a shipment is on its way, both customers and suppliers don’t always get clear updates about where it is or when it will arrive. This can lead to delays, missed deliveries, and unhappy customers.

On top of that, the process of returning products or handling complaints is often complicated and frustrating. Customers may have trouble requesting returns or refunds, and suppliers may find it hard to manage those requests properly. These problems cause stress for both sides and can hurt a business’s reputation.

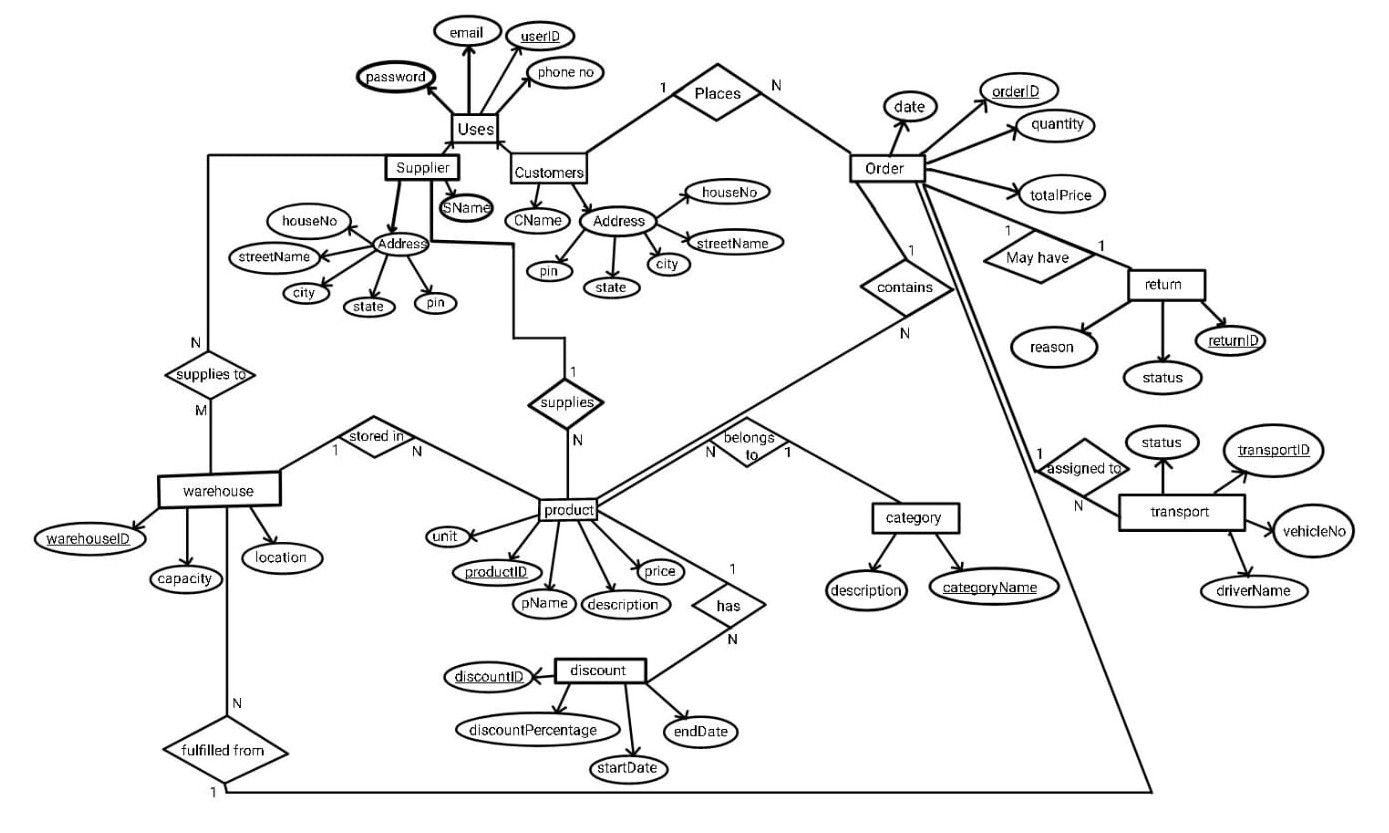
All of these issues, slow order handling, poor shipment tracking, and hard-to-manage returns—make the whole system less efficient. They also increase costs for the business, since more time and effort are needed to fix these problems. In the end, customers may become unhappy and look for better service elsewhere.

**3.2 Objectives**

* Streamline Operations: Combine all wholesale business activities, like orders, shipments, and returns into one easy-to-use platform.
* Enhance User Experience: Provide real-time tracking and notifications to keep users updated at every step.
* Improve Efficiency: Save time and reduce errors by automating tasks and improving communication between suppliers and customers.

**Chapter 4**

**Data Design**

* 1.  **ER Design**

**4.1.1 Entities**

* + - 1. The User entity represents individuals registered in the system. It acts as the base entity for generalized roles such as Supplier and Customer. Each user has a unique identifier (userID) and authentication attributes like password.
      2. Supplier: The Supplier entity inherits attributes from User and stores additional details specific to suppliers, such as their name (sName), contact information, and address.
      3. Customer: Like Supplier, the Customer entity inherits attributes from User and includes customer-specific details such as their name (cName), contact information, and address.
      4. Product: The Product entity represents items available for purchase. Each product is uniquely identified by productID, and it has attributes such as pName, description, price, and unit.
      5. Category: The Category entity groups products under specific classifications (categoryName), with a description to provide additional context.
      6. Order: The Order entity represents purchase orders placed by customers. It includes attributes such as orderID, date, quantity, and totalPrice.
      7. Return: The Return entity represents instances where customers return products. It includes attributes such as returnID, reason, status, and the associated orderID.
      8. Transport: The Transport entity tracks delivery logistics, including vehicles and drivers assigned to orders. Attributes such as transportID, vehicleNo, driverName, and status ensure detailed shipment tracking.
      9. Warehouse: The Warehouse entity represents locations where products are stored. Attributes such as warehouseID, location, and capacity define the storage facilities.
      10. Discount: The Discount entity represents promotional offers on products. Attributes such as discountID, discountPercentage, startDate, and endDate link discounts to specific products.
  1. **Reduction**
     1. **Mapping entities to relations**

Each entity is converted into a relation with attributes and primary keys.

|  |  |  |
| --- | --- | --- |
| Entity | Attributes | Primary Key |
| User | userID, password | userID |
| Supplier | email, phoneNo, address(houseNo, streetName, city, state, pin), sName | Foreign key userID from User |
| Customer | email, phoneNo, address(houseNo, streetName, city, state, pin), cName | Foreign key userID from User |
| Product | productID, pName, description, price, unit | productID |
| Category | categoryName, description | categoryName |
| Order | orderID, date, quantity, totalPrice | orderID |
| Return | returnID, reason, status | returnID |
| Transport | transportID, vehicleNo, driverName, status | transportID |
| Warehouse | warehouseID, location, capacity | warehouseID |
| Discount | discountID, discountPercentage, startDate, endDate | discountID |

***Table 1: Entities***

* + 1. **Handling generalization**

The ‘user’ entity is generalized into ‘supplier’ and ‘customer’.

Supplier and customer inherit userID from user.

|  |  |  |
| --- | --- | --- |
| Entity | Attributes | Primary Key |
| User | userID, password | userID |
| Supplier | email, phoneNo, address(houseNo, streetName, city, state, pin), sName, userID | Foreign key userID from User |
| Customer | email, phoneNo, address(houseNo, streetName, city, state, pin), cName, userID | Foreign key userID from User |

***Table 2: Generalisation***

* + 1. **Converting relationships**

The following relationships are present in the ER diagram,

|  |  |  |
| --- | --- | --- |
| Relationship | Type | Implementation |
| Customer-order | One to many | Customer places order |
| Order-product | Many to many | Order contains product |
| Order-return | One to one | Order may have return |
| Order-transport | One to many | Order assigned to transport |
| Product-category | Many to one | Product belongs to category |
| Supplier-product | One to many | Supplier supplies product |
| Product-discount | One to many | Product has discount |
| Supplier-warehouse | Many to many | Supplier supplies to warehouse |
| Warehouse-product | One to many | Product stored in warehouse |
| Warehouse-order | Many to one | Order fulfilled from warehouse |

***Table 3: table displaying relationship between the entities***

One to many: foreign keys are used to maintain references

Many to many: join tables are created

Updated list of tables,

|  |  |  |
| --- | --- | --- |
| Entity | Attributes | Primary Key |
| User | userID, password | userID |
| Supplier | email, phoneNo, address(houseNo, streetName, city, state, pin), sName, userID(references user) | Foreign key userID from User |
| Customer | email, phoneNo, address(houseNo, streetName, city, state, pin), cName, userID(references user) | Foreign key userID from User |
| Product | productID, pName, description, price, unit, categoryName(references category), userID(references supplier), warehouseID(references warehouse), orderID(references order) | productID |
| Category | categoryName, description | categoryName |
| Order | orderID, date, quantity, totalPrice, userID(references customer) | orderID |
| Return | returnID, reason, status, orderID(references order) | returnID |
| Transport | transportID, vehicleNo, driverName, status, orderID(references order) | transportID |
| Warehouse | warehouseID, location, capacity, orderID(references order) | warehouseID |
| Discount | discountID, discountPercentage, startDate, endDate, productID(references product) | discountID |
| Supplies | userID(references supplier), warehouseID(references warehouse) | userID, warehouseID |

***Table 4: Final Table of Entities***

* + 1. **Handling Composite and Multivalued Attributes**

Address is a composite attribute.

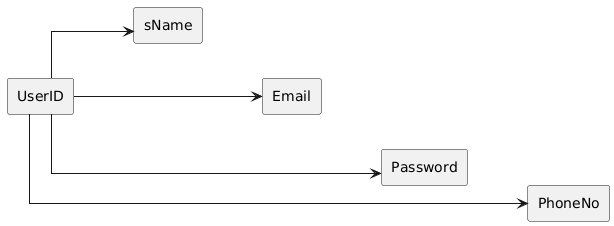
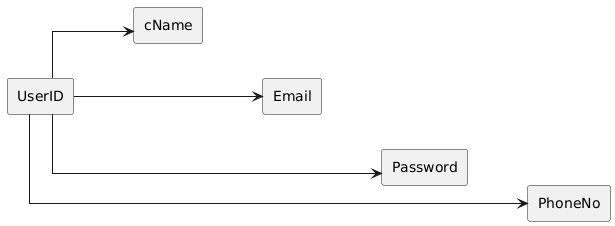
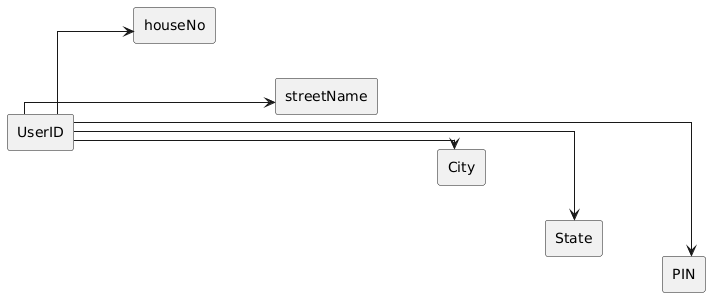
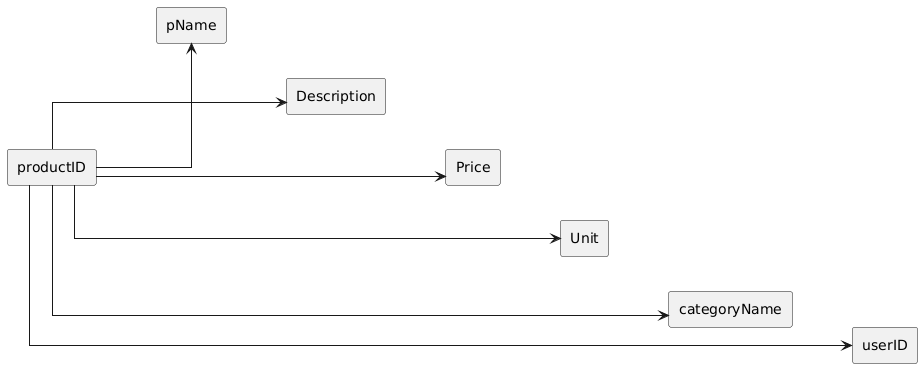
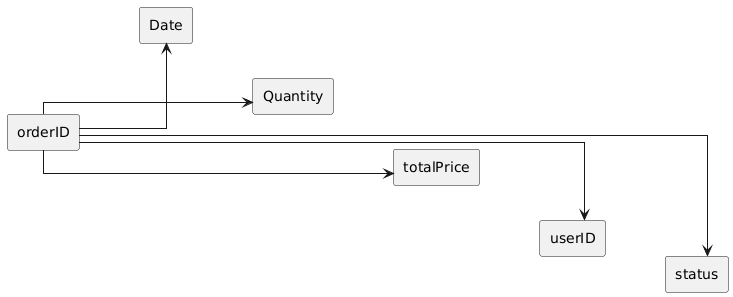
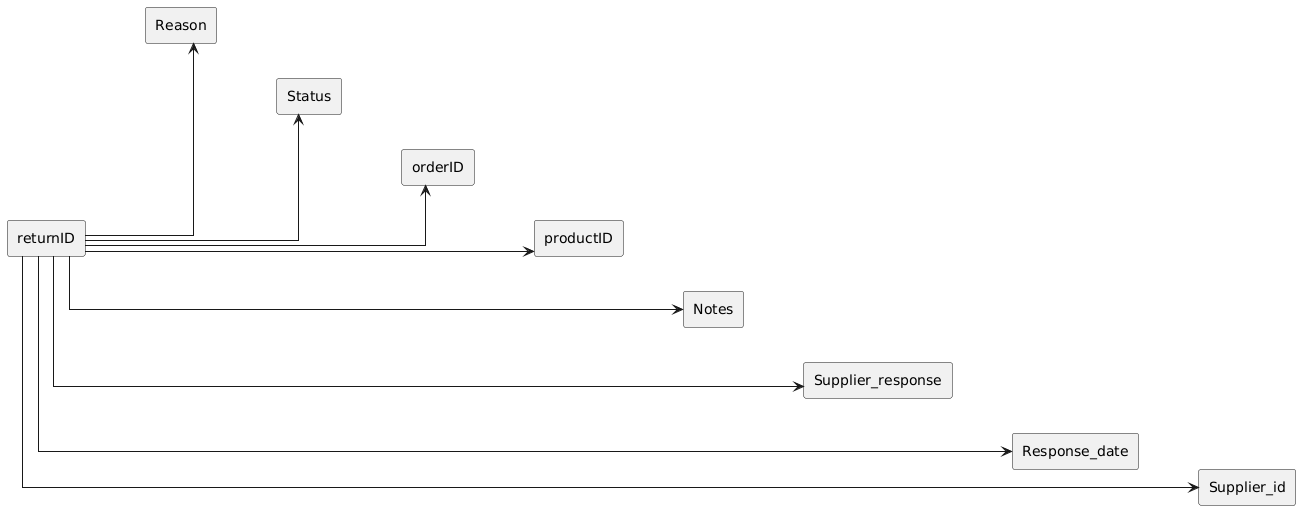
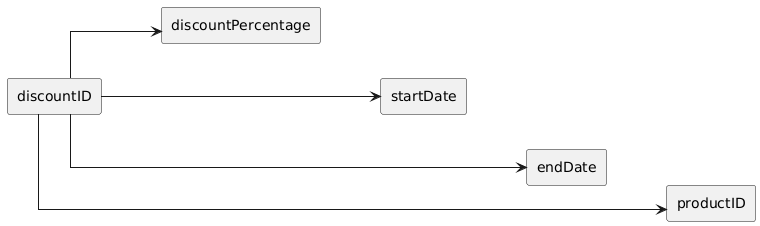
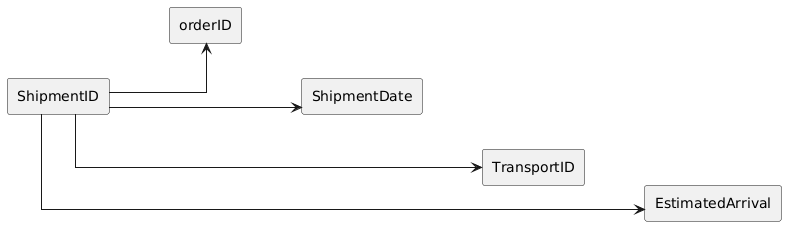
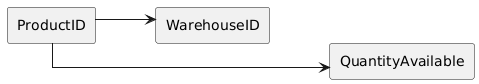
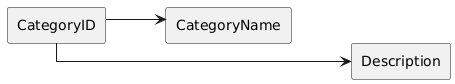
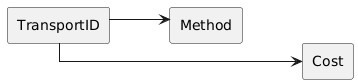
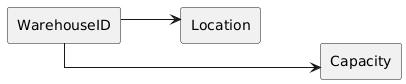
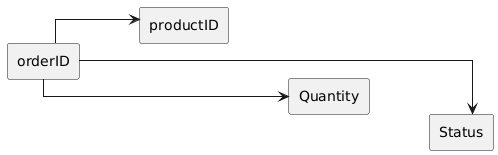
The address attribute was decomposed into atomic fields (houseNo, city, etc.) to enable location-based analytics and comply with 1NF.

|  |  |  |
| --- | --- | --- |
| Entity | Attributes | Primary Key |
| User | userID, password | userID |
| Supplier | userID, sName, email, phoneNo | userID |
| Customer | userID, cName, email, phoneNo | userID |
| Address | userID, houseNo, streetName, city, state, pin | userID |

***Table 5: Handling Multivalued Attributes***

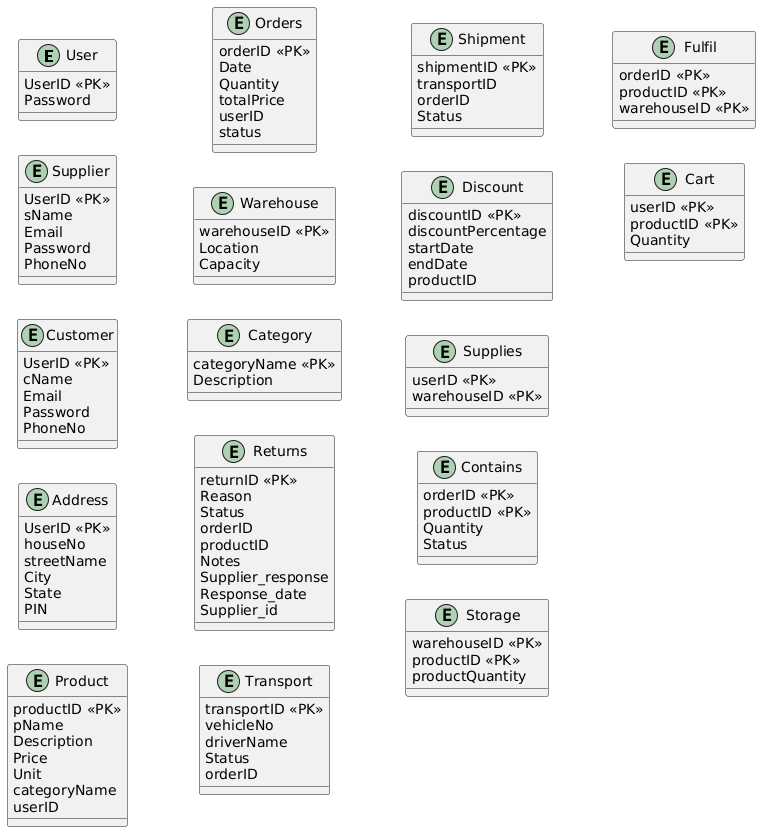
* 1. **Normalization-Shravani**
     1. **Dependency tables**
        1. **User table:**

PlantUML diagram

* + - 1. **Supplier table:** 
      2. **Customer table:** 
      3. **Address table:** 
      4. **Product table:** 
      5. **Order table:** 
      6. **Returns table:** 
      7. **Discount table:** 
      8. **Shipment table:** 
      9. **Storage table:** 
      10. **Category table:** 
      11. **Transport table:** 
      12. **Warehouse table:** 
      13. **Supplies table:** PlantUML diagram
      14. **Contains table:** 
      15. **Storage table:** 
      16. **Fulfil table:** 
    1. **First Normal Form (1NF)**

Ensure atomic values (no multivalued attributes or repeating groups).

If warehouseID and orderID are stored in the product table (according to reduction rules) it will need to unnecessary repeating of details. A similar problem occurs with orderID being stored in the warehouse table to avoid it, the following changes can be made:

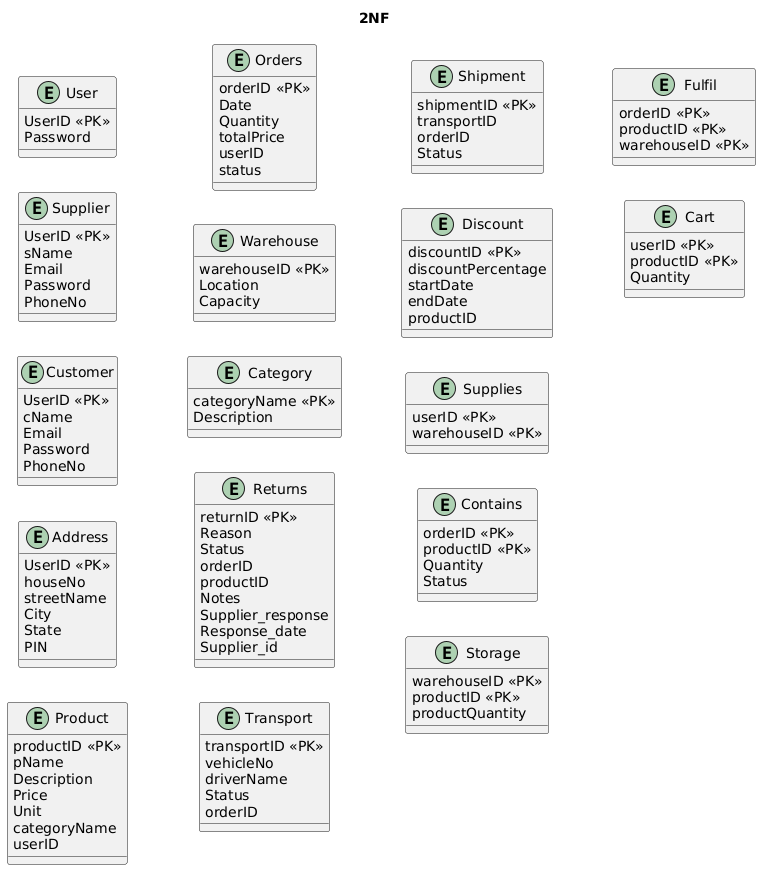


|  |  |  |
| --- | --- | --- |
| Entity | Attributes | Primary Key |
| Product | productID, pName, description, price, unit, categoryName(references category), userID(references supplier) | productID |
| Order | orderID, date, quantity, totalPrice, userID(references customer) | orderID |
| Warehouse | warehouseID, location, capacity | warehouseID |
| Contains | orderID(references order), productID(references product), productQuantity | orderID, productID |
| Storage | warehouseID(references warehouse), productID(references product), productQuantity | warehouseID, productID |
| Fulfil | warehouseID(references warehouse), orderID(references order), productID(references contains) | warehouseID, orderID, productID |

***Table 6: Tables after 1NF***

* + 1. **Second Normal Form (2NF)**

Remove partial dependencies. Partial dependencies were eliminated by ensuring all non-key attributes depend entirely on primary keys.



* + 1. **Third Normal Form (3NF)**

Remove transitive dependencies. Transitive dependencies were eliminated by ensuring all non-key attributes depend entirely on primary keys.



* 1. **Relational Schema**

These are the final relations-

|  |  |  |
| --- | --- | --- |
| Attribute | Data type | Constraints |
| UserID | VARCHAR (10) | Primary key |
| Password | VARCHAR (20) | NOT NULL |

***Table 7: User table schema***

|  |  |  |
| --- | --- | --- |
| Attribute | Data type | Constraints |
| UserID | VARCHAR (10) | Primary key, Foreign key references **User** |
| sName | VARCHAR(20) | NOT NULL |
| Email | VARCHAR(50) | NOT NULL |
| Password | VARCHAR (20) | NOT NULL |
| PhoneNo | INT | NOT NULL |

***Table 8: Supplier table schema***

|  |  |  |
| --- | --- | --- |
| Attribute | Data type | Constraints |
| UserID | VARCHAR (10) | Primary key, Foreign key references **User** |
| cName | VARCHAR(20) | NOT NULL |
| Email | VARCHAR(50) | NOT NULL |
| Password | VARCHAR (20) | NOT NULL |
| PhoneNo | INT | NOT NULL |

***Table 9: Customer table schema***

|  |  |  |
| --- | --- | --- |
| Attribute | Data type | Constraints |
| UserID | VARCHAR (10) | Primary key, Foreign key references **User** |
| houseNo | VARCHAR (10) | NOT NULL |
| streetName | VARCHAR (50) | NOT NULL |
| City | VARCHAR (50) | NOT NULL |
| State | VARCHAR (50) | NOT NULL |
| PIN | INT | NOT NULL |

***Table 10: Address table schema***

|  |  |  |
| --- | --- | --- |
| Attribute | Data type | Constraints |
| productID | VARCHAR (10) | Primary key |
| pName | VARCHAR(50) | NOT NULL |
| Description | VARCHAR (100) | NOT NULL |
| Price | INT | NOT NULL |
| Unit | VARCHAR(10) | NOT NULL |
| categoryName | VARCHAR (50) | Foreign key references **category** |
| userID | VARCHAR (10) | Foreign key references **Supplier** |

***Table 11: Product table schema***

|  |  |  |
| --- | --- | --- |
| Attribute | Data type | Constraints |
| orderID | VARCHAR (10) | Primary key |
| Date | DATE | NOT NULL |
| Quantity | INT | NOT NULL |
| totalPrice | INT | NOT NULL |
| userID | INT | Foreign key references **customer** |
| status | VARCHAR (20) | DEFAULT ‘Pending’ |

***Table 12: Orders table schema***

|  |  |  |
| --- | --- | --- |
| Attribute | Data type | Constraints |
| warehouseID | VARCHAR (10) | Primary key |
| Location | VARCHAR (50) | NOT NULL |
| Capacity | INT | NOT NULL |

***Table 13: Warehouse table schema***

|  |  |  |
| --- | --- | --- |
| Attribute | Data type | Constraints |
| categoryName | VARCHAR (30) | Primary key |
| Description | VARCHAR (50) | NOT NULL |

***Table 14: Category table schema***

|  |  |  |
| --- | --- | --- |
| Attribute | Data type | Constraints |
| returnID | VARCHAR (10) | Primary key |
| Reason | VARCHAR (100) | NOT NULL |
| Status | VARCHAR (10) | DEFAULT ‘Pending’ |
| orderID | VARCHAR (10) | Foreign key references **order** |
| productID | VARCHAR (10) | Foreign key references **product** |
| Notes | TEXT | DEFAULT NULL |
| Supplier\_response | TEXT | DEFAULT NULL |
| Response\_date | DATETIME | DEFAULT NULL |
| Supplier\_id | VARCHAR (10) | Foreign key references **supplier** |

***Table 15: Returns table schema***

|  |
| --- |
|  |

|  |  |  |
| --- | --- | --- |
| Attribute | Data type | Constraints |
| transportID | VARCHAR (10) | Primary key |
| vehicleNo | VARCHAR (10) | NOT NULL |
| driverName | VARCHAR (20) | NOT NULL |
| Status | VARCHAR (10) | NOT NULL |
| orderID | VARCHAR (10) | Foreign key references **orders** |

***Table 16: Transport table schema***

|  |  |  |
| --- | --- | --- |
| Attribute | Data type | Constraints |
| shipmentID | VARCHAR (10) | Primary key |
| transportID | VARCHAR (10) | Foreign key references **transport** |
| orderID | VARCHAR (10) | Foreign key references **orders** |
| Status | Enum (‘Pending’, ‘InTransit’ ,’ delivered’) | DEFAULT ‘Pending’ |

***Table 17: Shipment table schema***

|  |  |  |
| --- | --- | --- |
| Attribute | Data type | Constraints |
| discountID | VARCHAR (10) | Primary key |
| discountPercentage | INT | NOT NULL |
| startDate | DATE | NOT NULL |
| endDate | DATE | NOT NULL |
| productID | VARCHAR (10) | Foreign key references **product** |

***Table 18: Discount table schema***

|  |  |  |
| --- | --- | --- |
| Attribute | Data type | Constraints |
| userID | VARCHAR (10) | Primary key |
| warehouseID | VARCHAR (10) | Primary key |

***Table 19: Supplies table schema***

|  |  |  |
| --- | --- | --- |
| Attribute | Data type | Constraints |
| orderID | VARCHAR (10) | Primary key |
| productID | VARCHAR (10) | Primary key |
| Quantity | INT | NOT NULL |
| Status | VARCHAR (10) | DEFAULT ‘Pending’ |

***Table 20: Contains table schema***

|  |  |  |
| --- | --- | --- |
| Attribute | Data type | Constraints |
| warehouseID | VARCHAR (10) | Primary key |
| productID | VARCHAR (10) | Primary key |
| productQuantity | INT | NOT NULL |

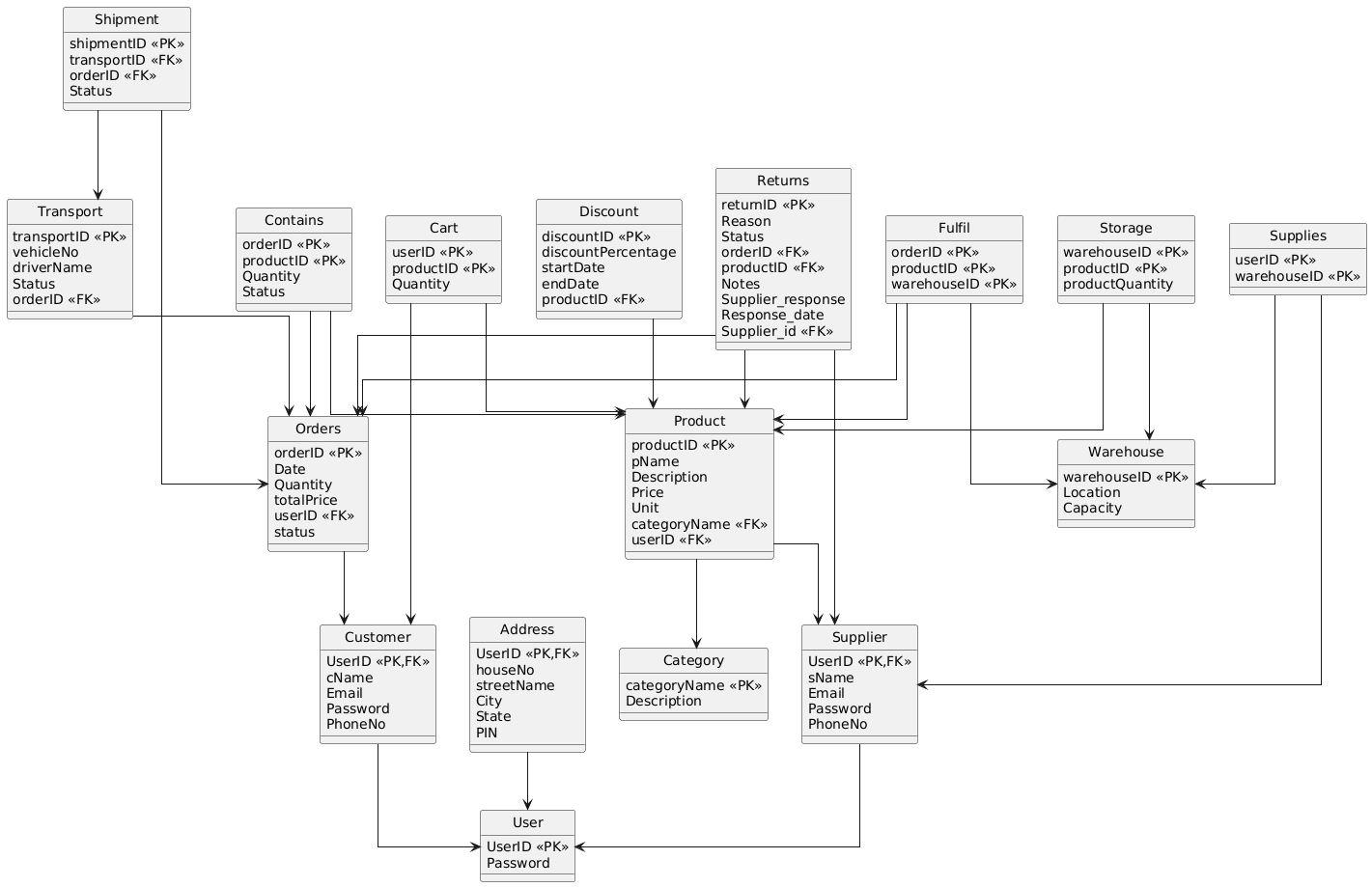
***Table 21: Storage table schema***

|  |  |  |
| --- | --- | --- |
| Attribute | Data type | Constraints |
| orderID | VARCHAR (10) | Primary key |
| productID | VARCHAR (10) | Primary key |
| warehouseID | VARCHAR (10) | Primary key |

***Table 22: Fulfil table schema***

|  |  |  |
| --- | --- | --- |
| Attribute | Data type | Constraints |
| userID | VARCHAR (10) | Primary key. Foreign key references **customer** |
| productID | VARCHAR (10) | Primary key. Foreign key references **product** |
| Quantity | INT | NOT NULL |

***Table 23: Cart table schema***



*Schema diagram*

**Chapter 5**

**Methodology**

**Implementation details with block diagram to explain the project in detail**

### **5.1 System Overview**

This project is a web-based wholesale management system named **MarketHub** designed to streamline wholesale transactions between suppliers and customers. The application covers a wide range of features including product listing, cart management, order placement and tracking, warehouse stock control, return handling, and transport management. Built with a modern tech stack (HTML, CSS, JavaScript, Flask, and MySQL), it provides dedicated interfaces for both **customers** and **suppliers** to enhance their interaction and operational efficiency.

**5.2 Architecture and Technology Stack**

|  |  |
| --- | --- |
| **Layer** | **Technology Used** |
| Frontend | HTML, CSS, JavaScript |
| Backend | Python with Flask framework |
| Database | MySQL (Relational DBMS) |
| Web Server | Flask Development Server |

The application follows a client-server architecture, where Flask handles the routing, logic, and DB communication.

The system architecture includes:

* User Interaction: Customers and suppliers interact with the front end.
* Backend Processing: Python Flask handles request and updates the database.
* Database Communication: MySQL stores and retrieves data.

**5.3 Module Descriptions**

#### 5.3.1 Login / Signup Module

* **Customer** or **Supplier** selects their role while signing up.
* Validates user credentials during login and checks user type (from Customer or Supplier table).
* Information stored in User, Customer, and Supplier tables.

#### 5.3.2 Product Browsing and Cart Module (Customer)

* Retrieves product list from Product and Category.
* Enables search, filter, and sort functionality.
* Adds selected items to Cart (stored in DB with userID, productID, quantity).

#### 5.3.3 Order Placement Module

* Converts cart items into an order (Orders, Contains).
* Calculates totalPrice, assigns orderID.
* Clears Cart on order confirmation.

#### 5.3.4 Order History & Tracking (Customer)

* Displays previous orders using Orders, Contains, and Shipment.
* Tracks status from Shipment and Transport.

#### 5.3.5 Returns and Complaints (Customer)

* Customers can file complaints via Returns table (linked to Orders).
* Complaints are sent to the supplier of the related product.

#### 5.3.6 Supplier Product Management

* Suppliers can add products (Product), categorized via Category.
* Products are stocked into specific warehouses (Storage, Supplies).

#### 5.3.7 Stock & Warehouse Management

* Displays available warehouses (Warehouse).
* Allows supplier to manage stock levels (Storage, Supplies).

#### 5.3.8 Order Fulfilment (Supplier)

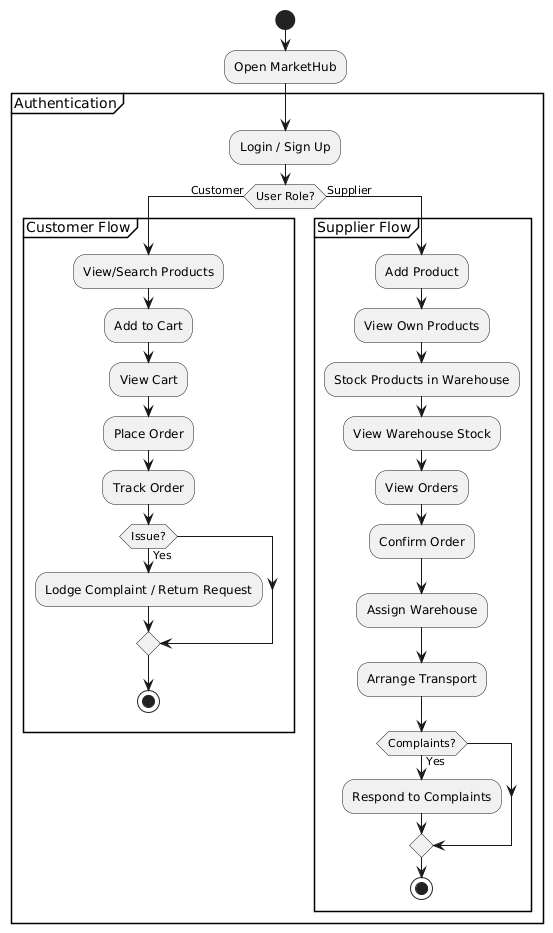
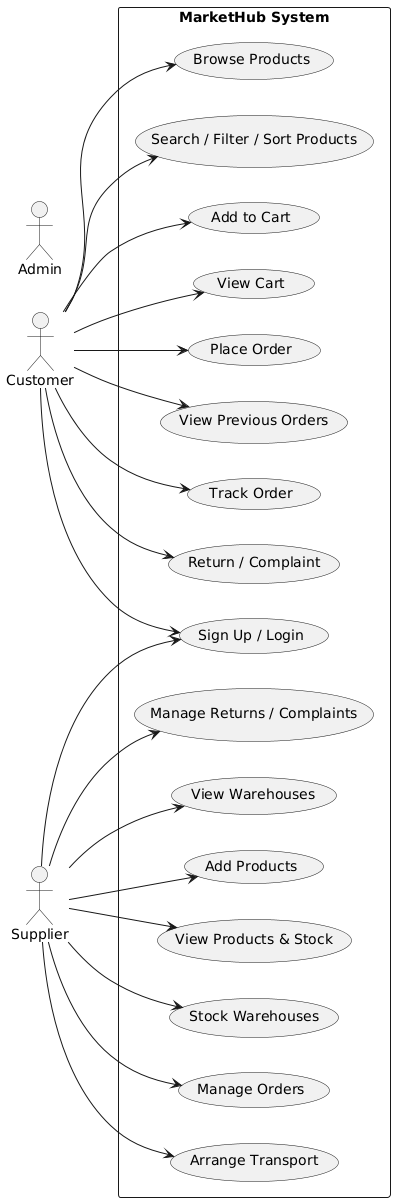
* Views pending orders and approves them (Orders).
* Arranges transport (Transport) and links it to orders.
* Updates shipment details (Shipment) and marks delivery stages.

#### 5.3.9 Complaint Handling (Supplier)

* Suppliers receive complaints (Returns) about their products.
* Can view and update complaint resolution status.

### **5.4 Assumptions and Limitations**

* **No Payment Gateway**: Transactions are not monetized.
* **No Session Handling**: Login info is not stored via sessions; userID is handled through file-based storage or similar.
* **Single User Type per Login**: A user cannot act as both customer and supplier.
* **Return Handling**: Manual complaint resolution without automated refund process.



**Chapter 6**

**Results**

## **6.1 Overview of the Developed System**

The developed system, MarketHub, is a complete wholesale business management platform designed to streamline and digitize essential processes such as ordering, shipment tracking, stock management, and return handling. It supports two primary user roles: **Customer** and **Supplier**, each with dedicated functionalities tailored to their needs.

Customers can browse and filter products, place orders, view past orders, track shipments in real-time, and raise complaints or returns. Suppliers, on the other hand, can add and manage products, update stock across warehouses, approve or reject customer orders, handle transportation logistics, and address customer complaints.

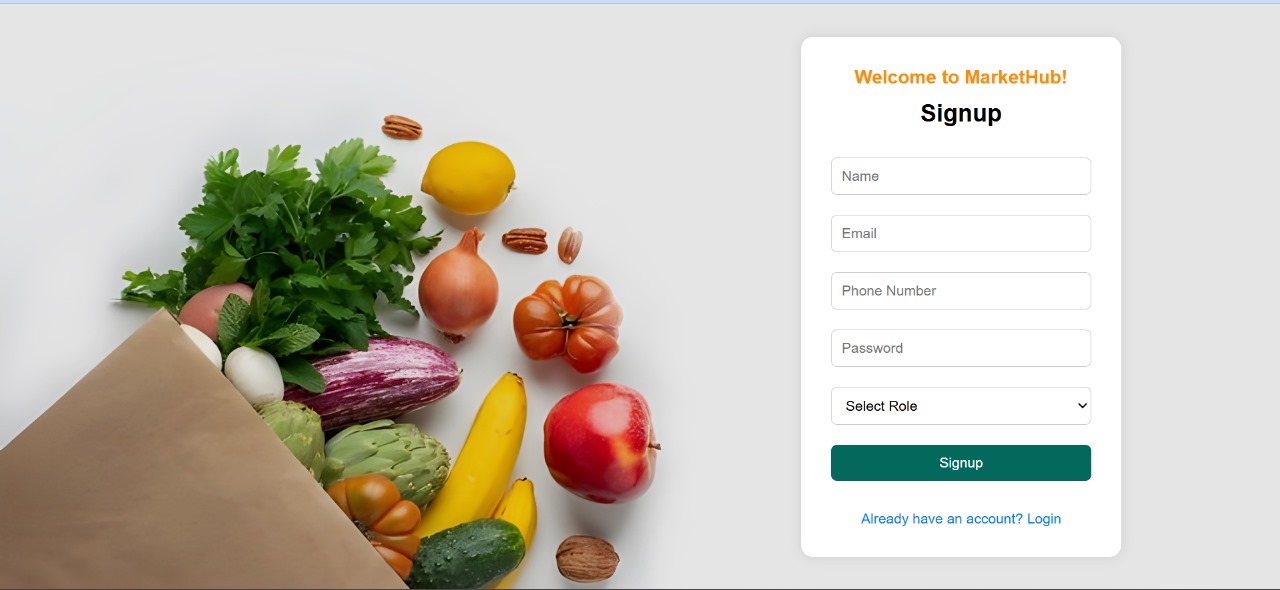
The system leverages HTML, CSS, JavaScript for the frontend, and Python (Flask) with MySQL for the backend and database integration.

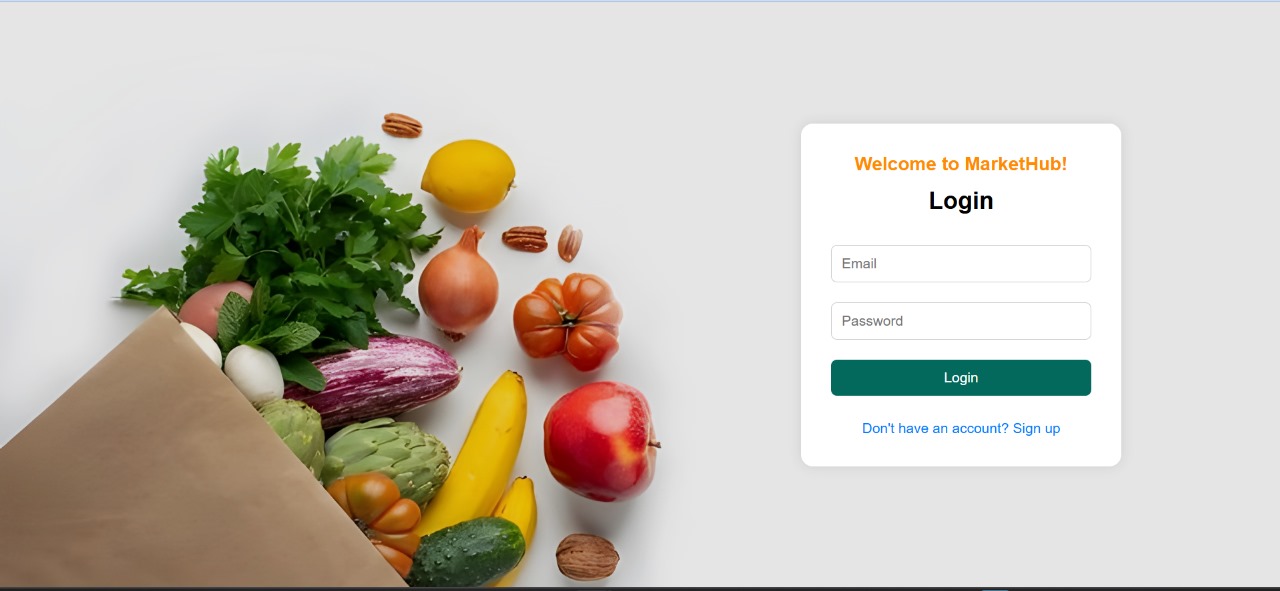
## **6.2 System Output and Interface Screenshots**

Below are key screenshots demonstrating the functionality and flow of the application.

### **Login and Signup**

Users can sign up as either a Customer or Supplier and securely log in. The system directs them to role-specific dashboards.

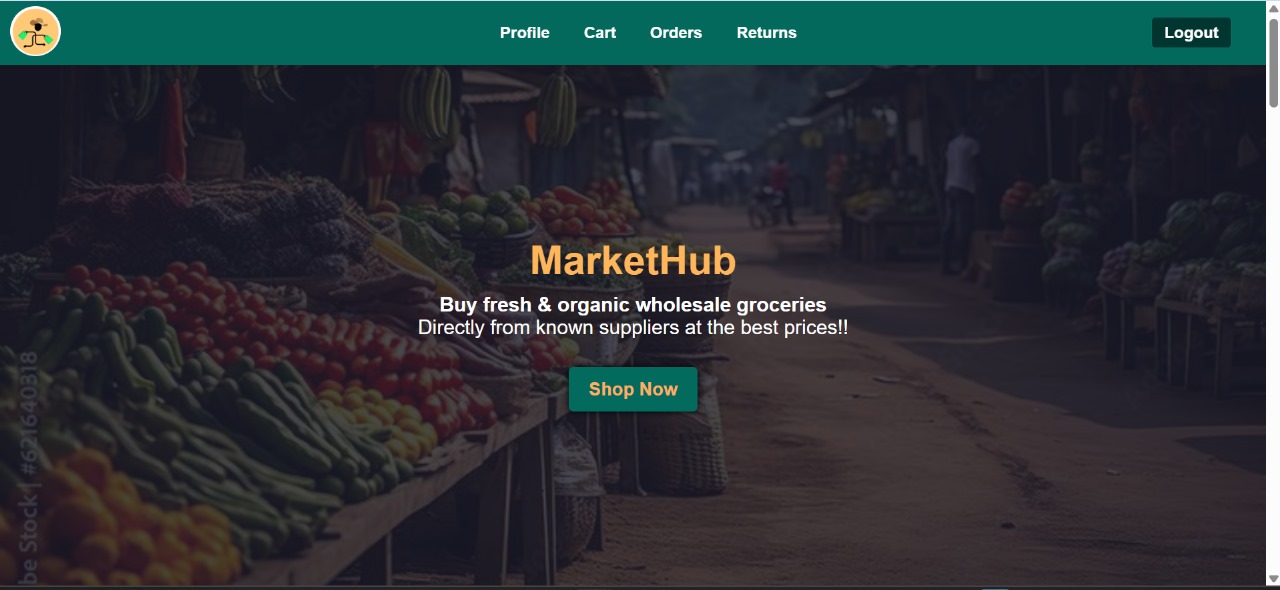


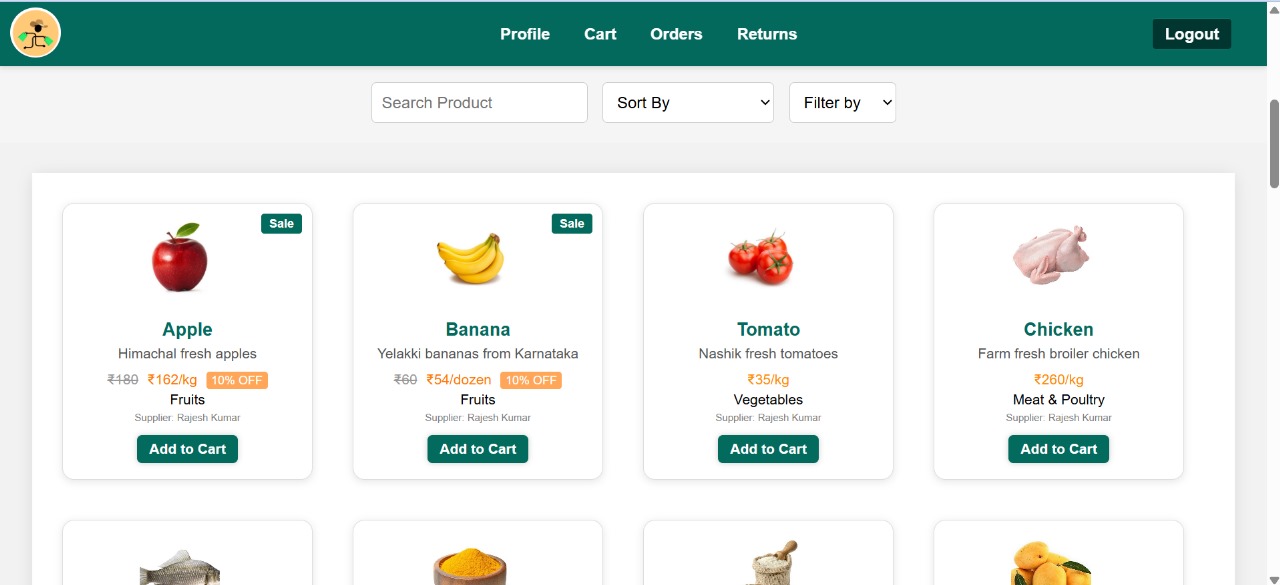


### **Customer Interface: Product Browsing and Ordering**

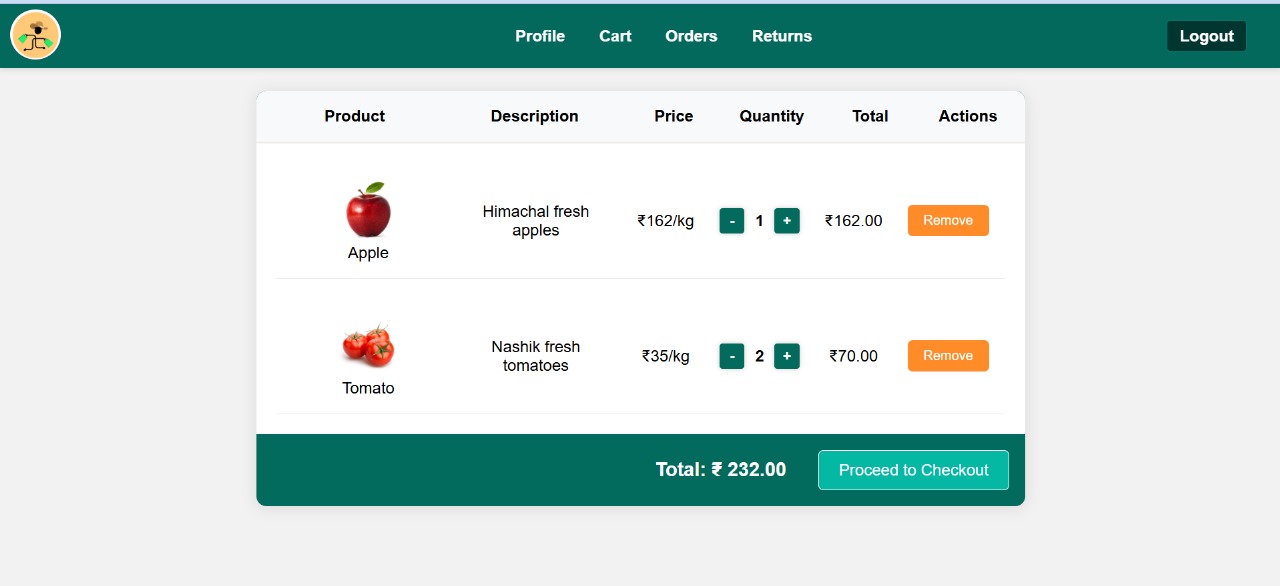
Customers can view a list of available products, filter or search them, and add items to a cart.

Customers can filter and sort products based on price, category, or name.





After adding products to the cart, customers can proceed to checkout, confirm the order, and track their purchases.

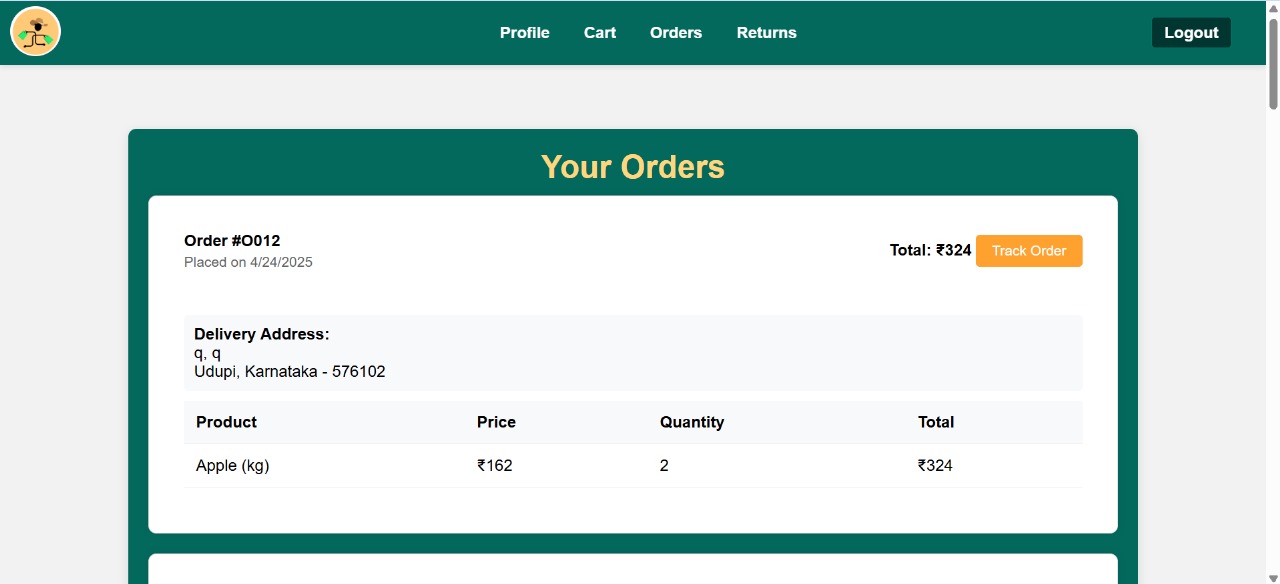


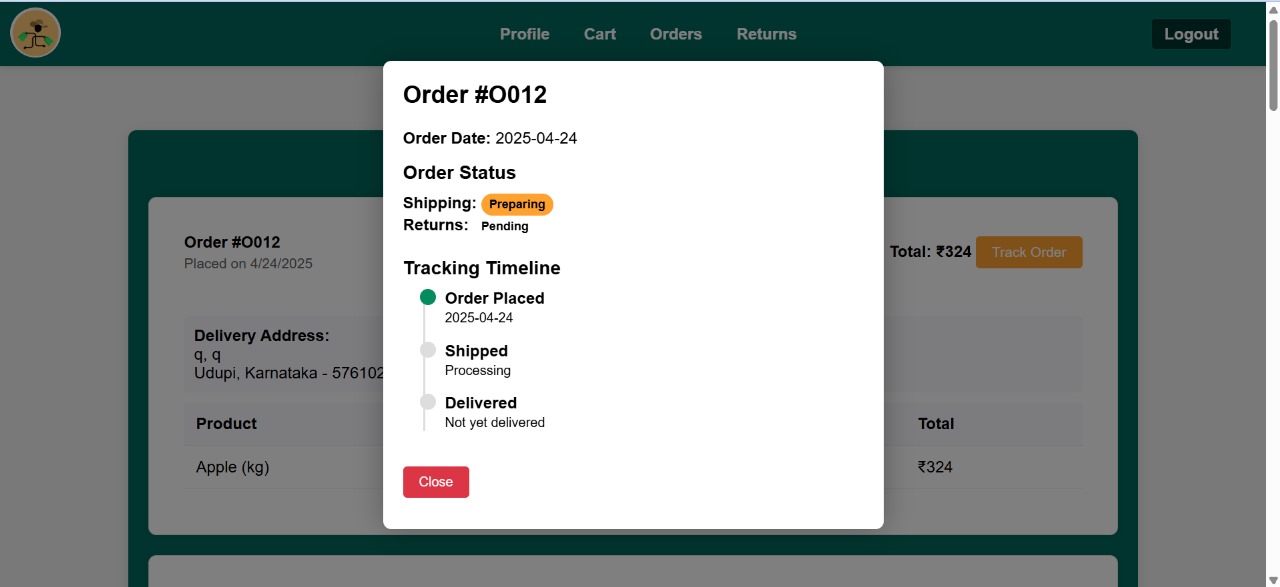
### **Order Tracking and Returns**

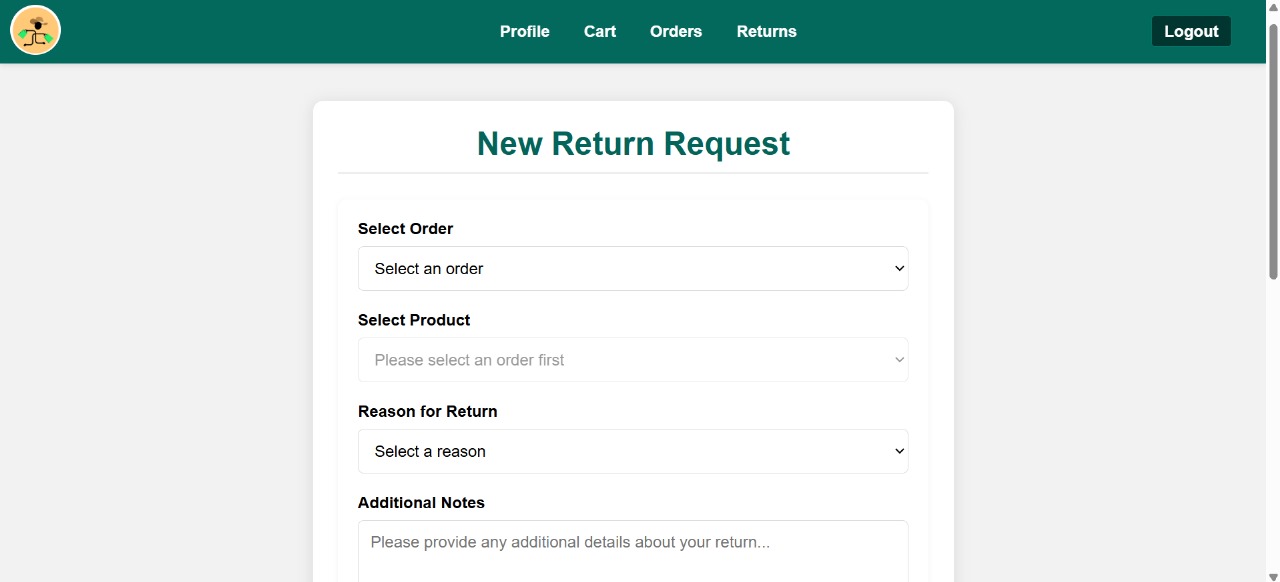
Customers can track their order status (Pending, In Transit, Delivered) and raise return/complaint requests for individual items.

Real-time order tracking with status updates from the shipment table.

Customers can file returns or complaints which are routed to the respective supplier for resolution.

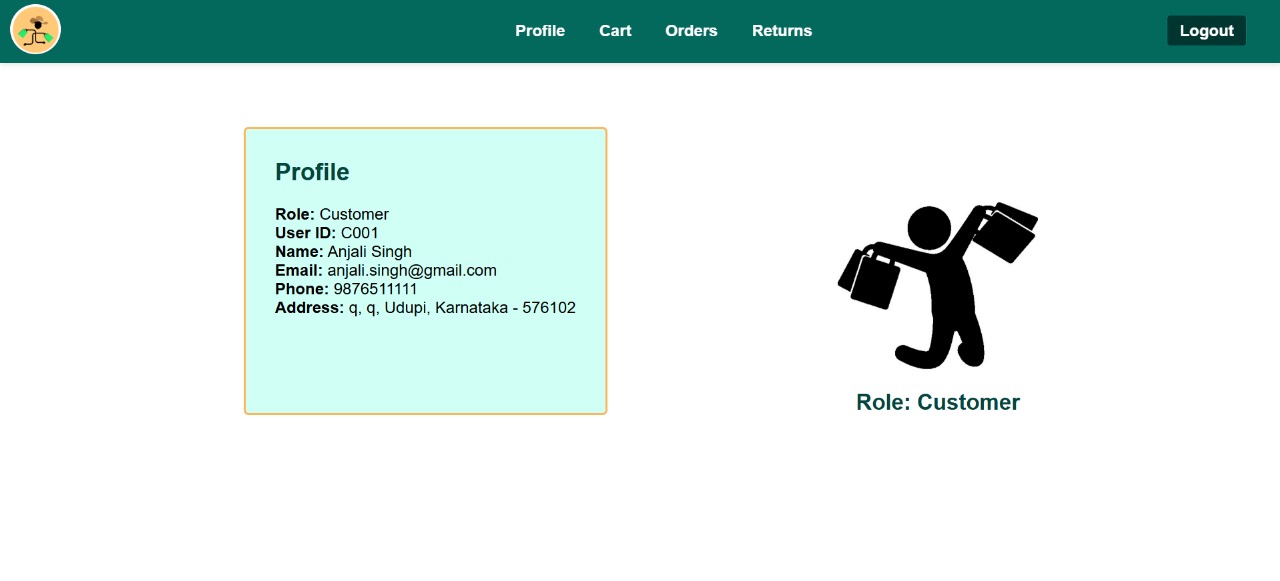






### **View Customer Profile**

Check out the customer details in customer profile



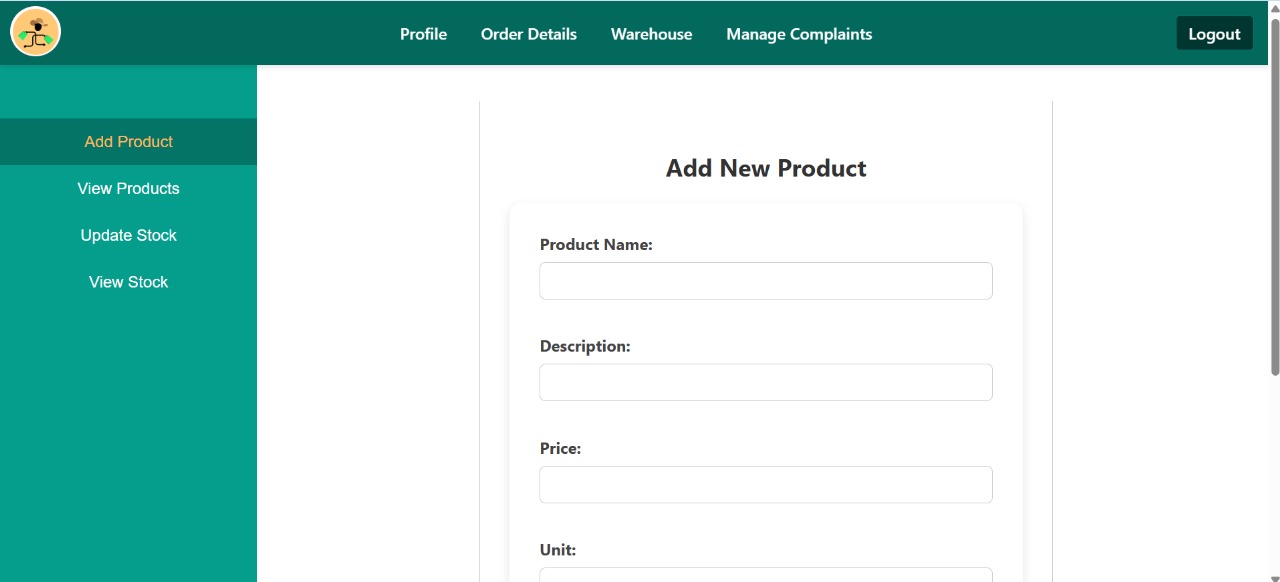
### **Supplier Interface**

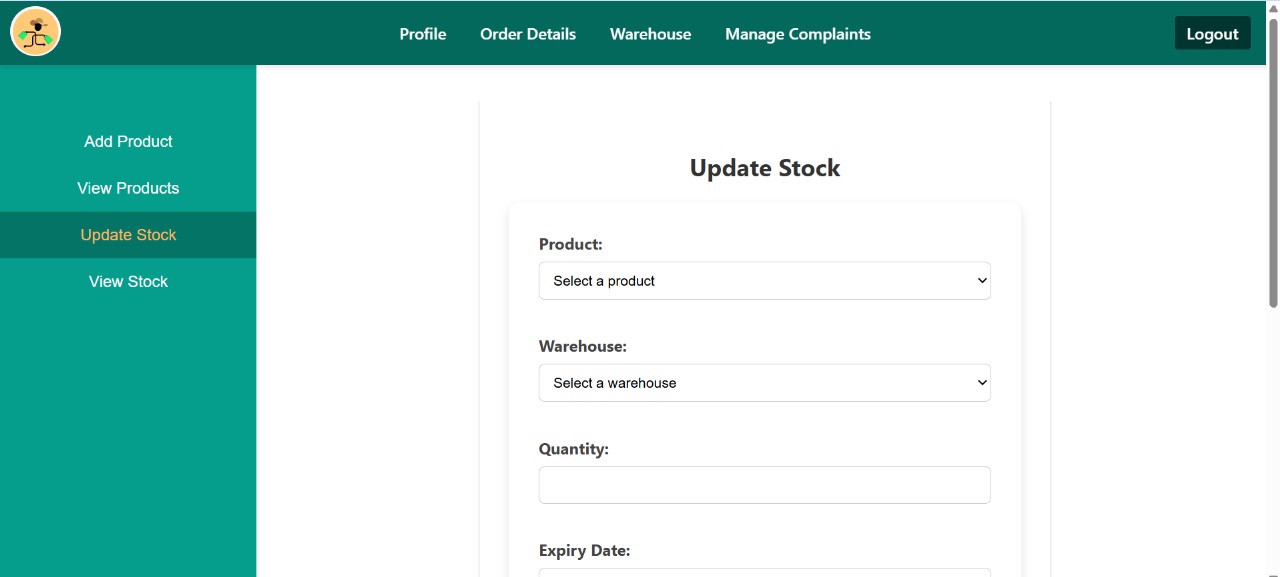
Suppliers can manage their product listings, assign stock to warehouses, fulfil orders, and arrange transportation.

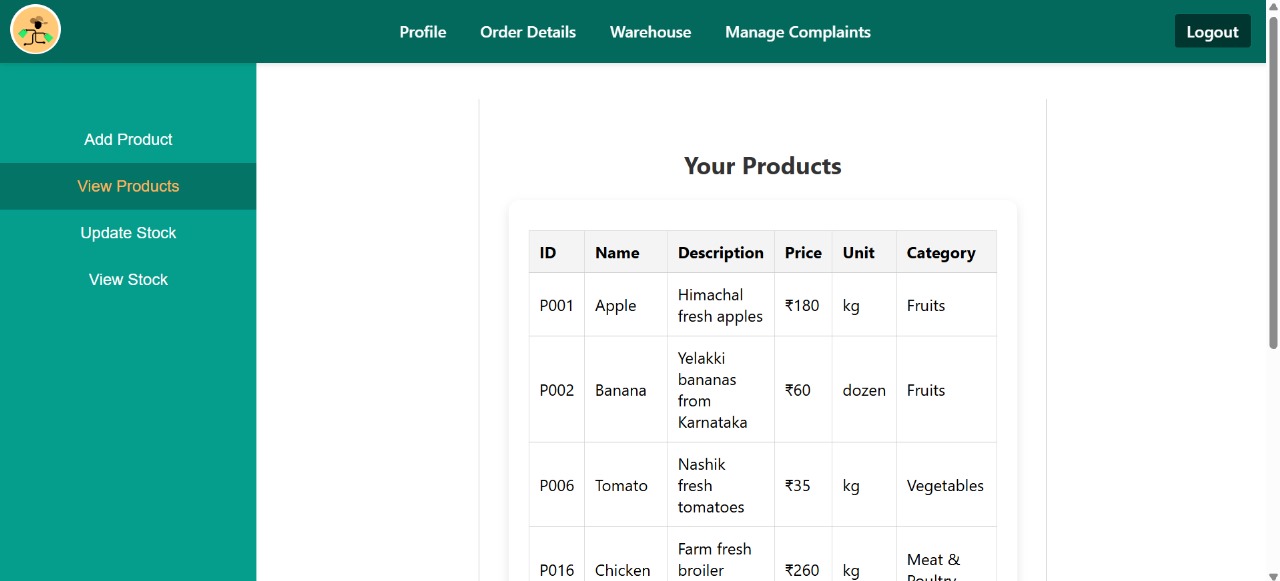
Interface for adding products and distributing them across warehouses.

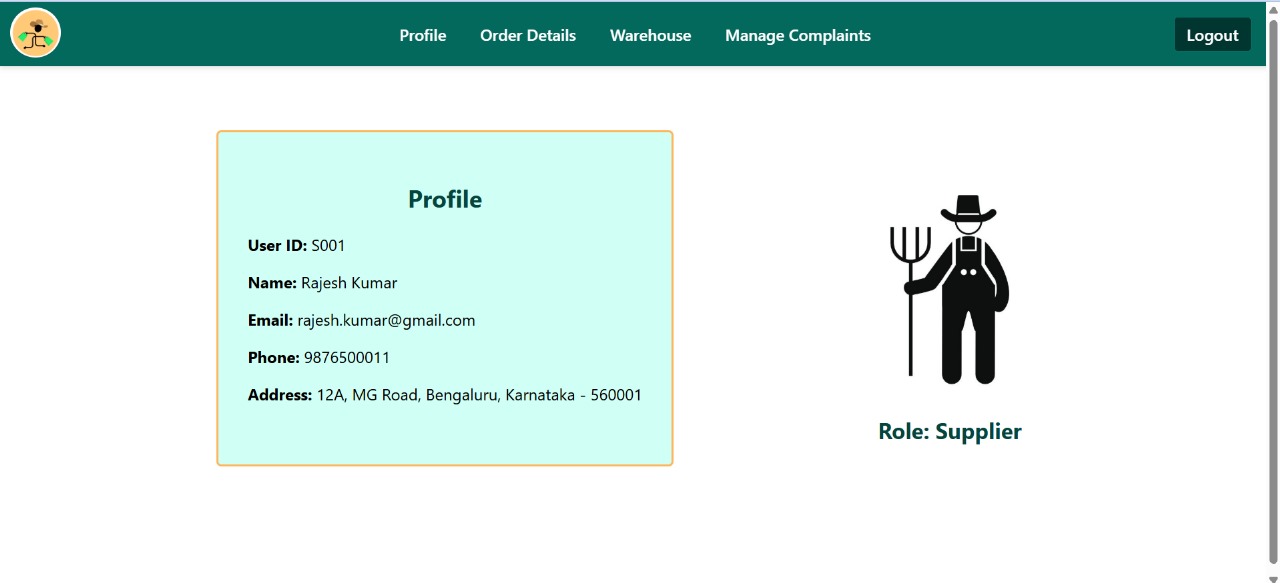
Supplier dashboard to view customer orders, approve or reject them, and assign transport for delivery.

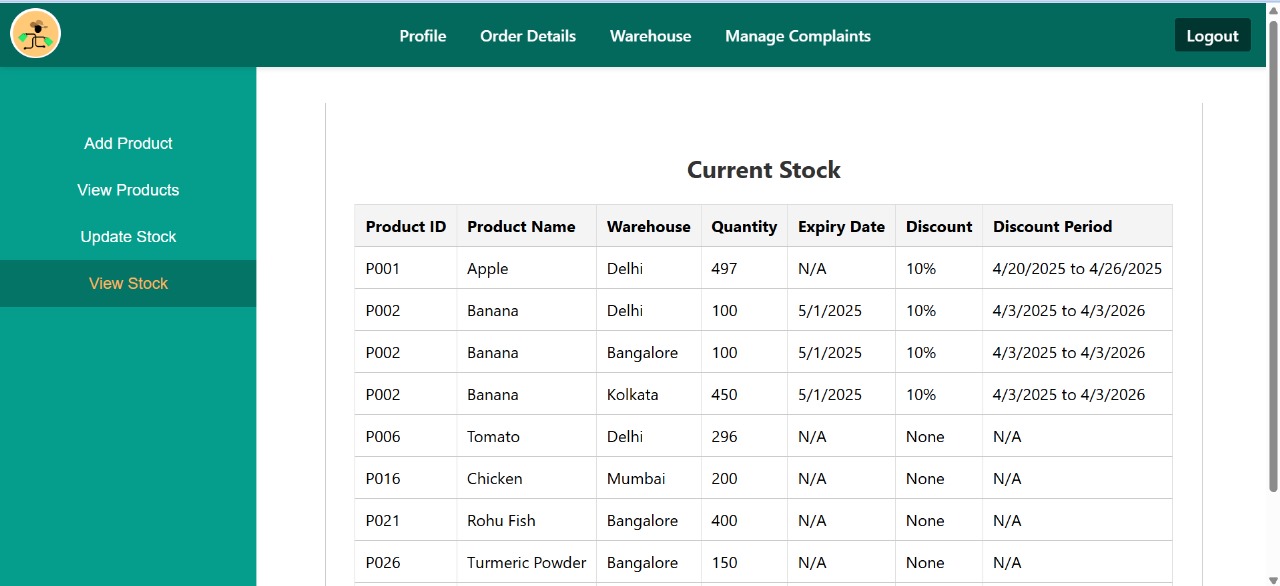
Suppliers can view all complaints related to their products and respond accordingly.

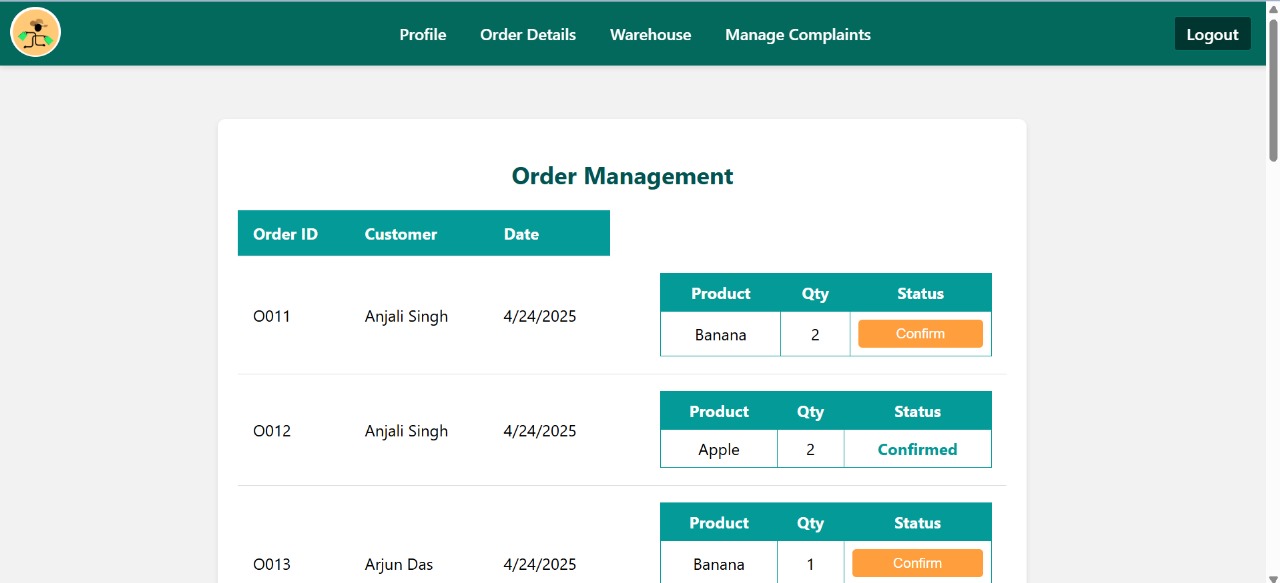


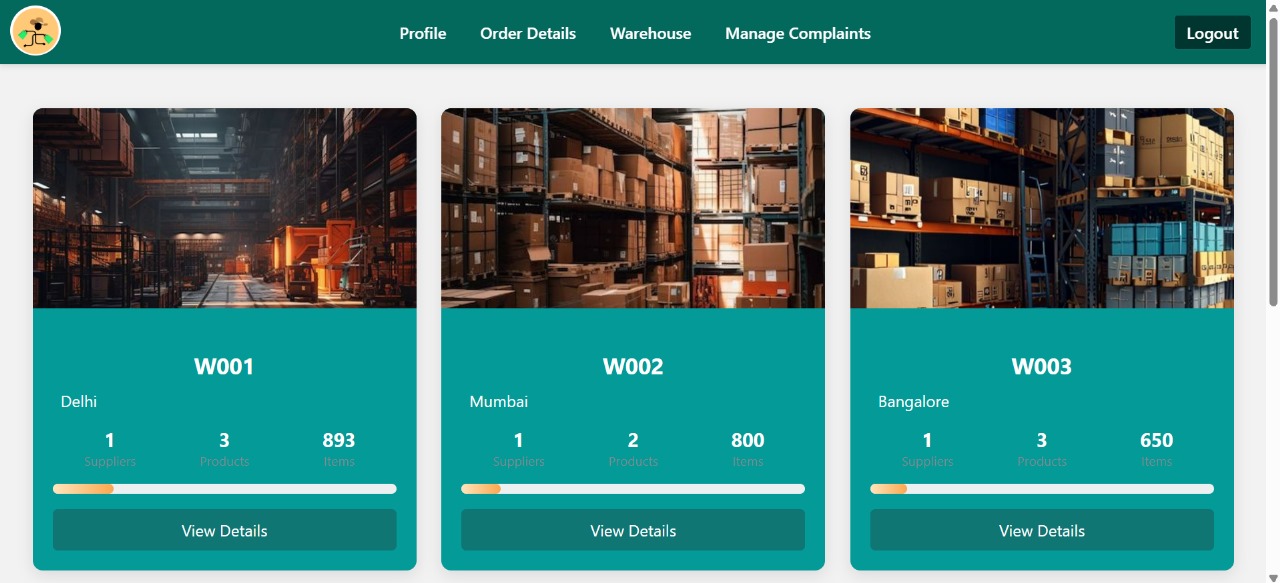


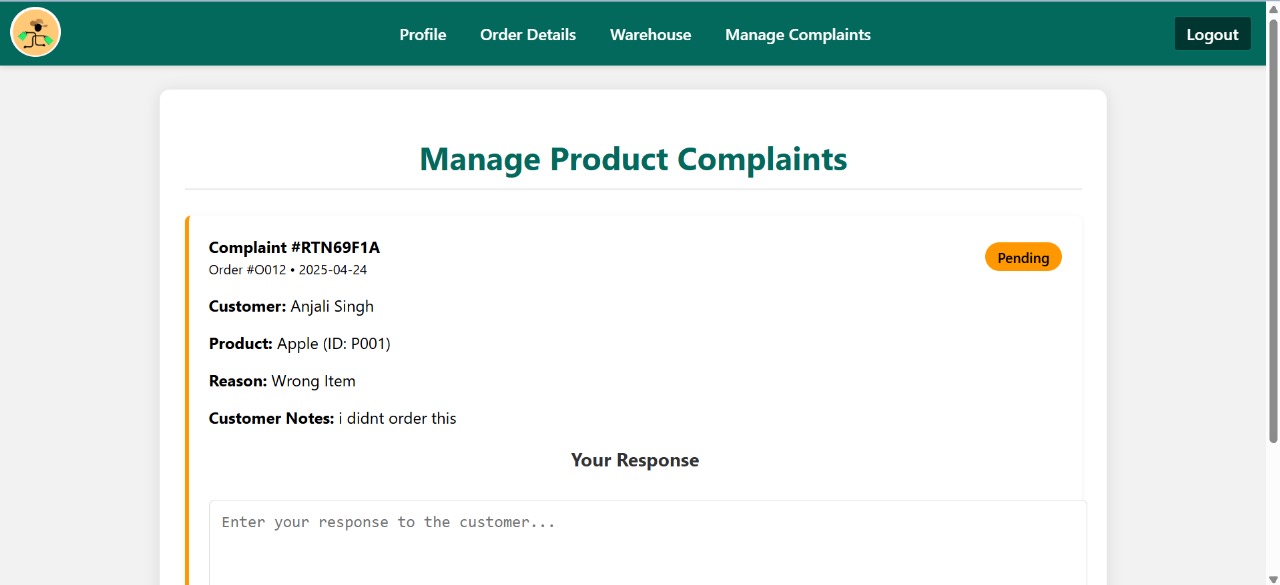












## **6.3 Testing and Validation**

All modules were manually tested for functionality and correctness:

* **Authentication**: Ensured correct login with role-based redirection.
* **Product Search/Filter**: Verified filtering by category and sorting by price/name.
* **Order Processing**: Orders reflect correct quantities and prices; cart is cleared post-checkout.
* **Returns**: Submissions logged correctly in the database and visible to suppliers.
* **Database Consistency**: Foreign key constraints and validations were checked manually.
* **Edge Cases Tested**:
  + Attempting to place an order with an empty cart
  + Adding invalid product quantities
  + Handling stock updates post-order

**6.4 Comparison with Objectives**

|  |  |
| --- | --- |
| **Objective** | Achievement |
| Streamline Operations | All core activities (orders, stock, transport, complaints) are centralized in one platform. |
| Enhance User Experience | Customers receive updates on order status and can interact through returns/complaints. |
| Improve Efficiency | Automation in stock and order tracking reduces manual work and improves communication. |

## **6.5 Limitations**

While the system is functional, a few limitations exist:

* **No session management**: User authentication lacks session tracking; any page refresh requires re-login.
* **No payment integration**: Orders are processed without payment gateways.
* **Basic UI**: The interface, though functional, can be improved using modern frameworks like Bootstrap or React.
* **No admin panel**: The system currently does not include administrative oversight or analytics.

**Chapter 7**

**Conclusion and Future Work**

MarketHub was created to solve common problems in wholesale businesses, like slow order management, poor shipment tracking, and difficult return processes. Bringing all important features into one platform, it made daily operations easier and faster for both suppliers and customers.

The system helped save time, reduced errors and made communication smoother. Customers were more satisfied because they could track their orders and get help quickly. Suppliers could manage their work better and focus on growing their business.

In conclusion, MarketHub proves that with the right tools, wholesale businesses can work more efficiently, keep customers happy, and achieve better results.

## **Scope for Future Work**

While the current system successfully addresses core functionalities, several enhancements can further improve usability, scalability, and automation:

* **Session-based Login**: Introduce persistent user sessions for improved security and smoother navigation.
* **Admin Dashboard**: Implement an admin module to oversee and manage users, warehouses, transport logistics, and system-wide analytics.
* **Payment Gateway Integration**: Enable secure payment processing directly through the platform.
* **Email/SMS Notification System**: Send real-time alerts to customers and suppliers regarding order status, shipment tracking, and returns.
* **Stock Alerts**: Automatically notify suppliers when inventory levels drop below a threshold.
* **Improved UI/UX**: Upgrade the interface using responsive frameworks like Bootstrap or React for a more modern user experience.
* **Advanced Filtering and Proximity-Based Ordering**: Allow customers to filter and prioritize products based on availability in nearby warehouses, and enable suppliers to allocate products efficiently to warehouses closest to demand zones.
* **Transport Management System**: Design a more comprehensive module for managing transportation logistics including vehicle availability, delivery optimization, and driver tracking.
* **Warehouse Management by Admin**: Provide an admin interface to monitor warehouse capacity, location, and efficiency to support better logistics planning and storage optimization.

**References**

[1] Python Software Foundation, Flask Documentation. [Online]. Available: <https://flask.palletsprojects.com/>

[2] Oracle Corporation, MySQL Documentation. [Online]. Available: <https://dev.mysql.com/doc/>

[3] IEEE, IEEE Guide to Software Requirements Specifications, IEEE Std 830-1998. [Online]. Available: <https://ieeexplore.ieee.org/document/720574>

[4] A. Silberschatz, H. F. Korth, and S. Sudarshan, Database System Concepts, 6th ed., New York, NY, USA: McGraw-Hill, 2010.

[5] R. Elmasri and S. B. Navathe, Fundamentals of Database Systems, 7th ed., Boston, MA, USA: Pearson, 2016.